Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Сортировки(Слиянием, подсчетом, быстрая или Хоара, блочная, Шелла)

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Мухаров Тимур Рустамович

Проверил

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Г. Пермь-2023

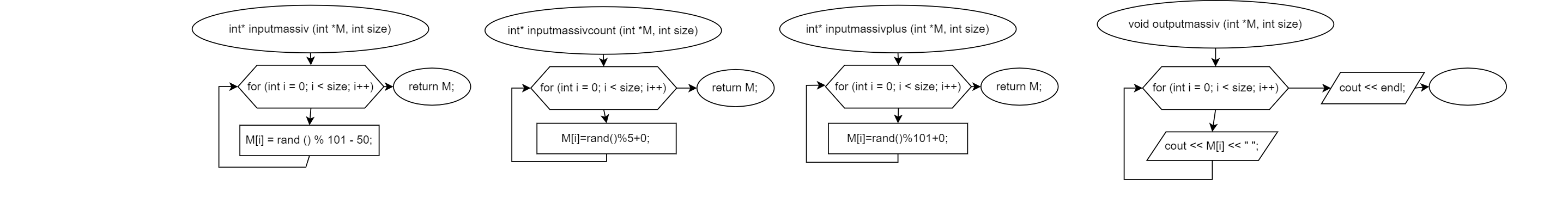
**Постановка задачи**

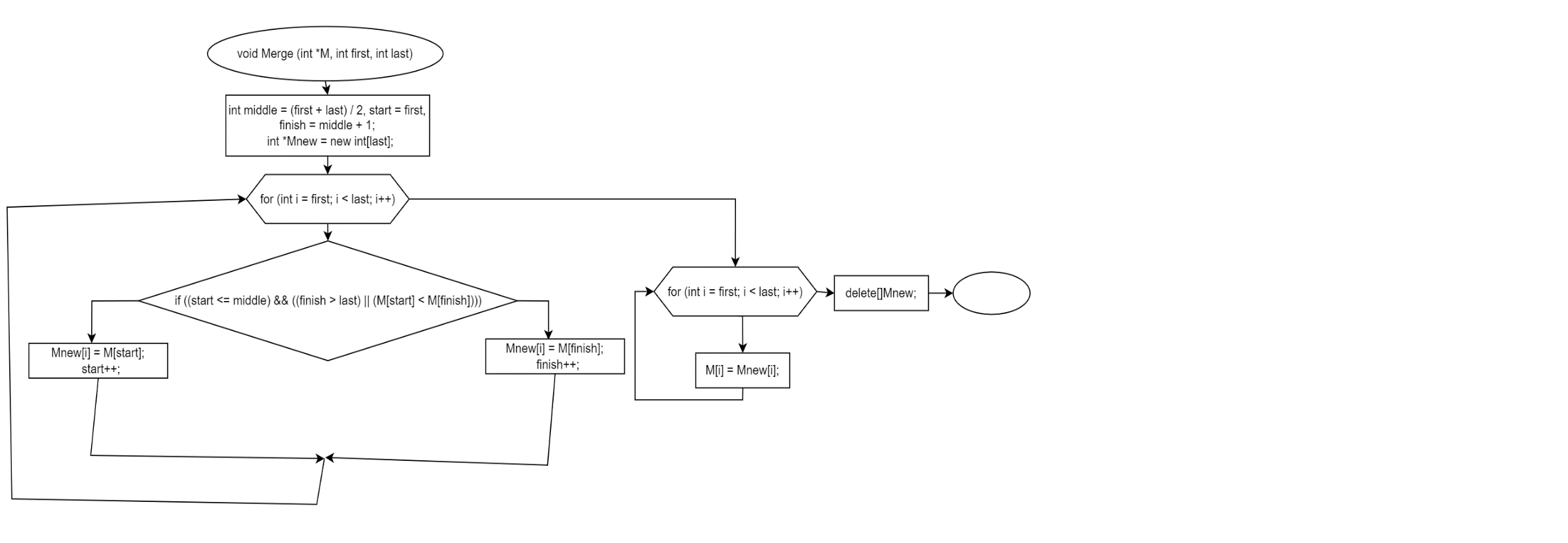
Рассмотреть какая сортировка быстрее.

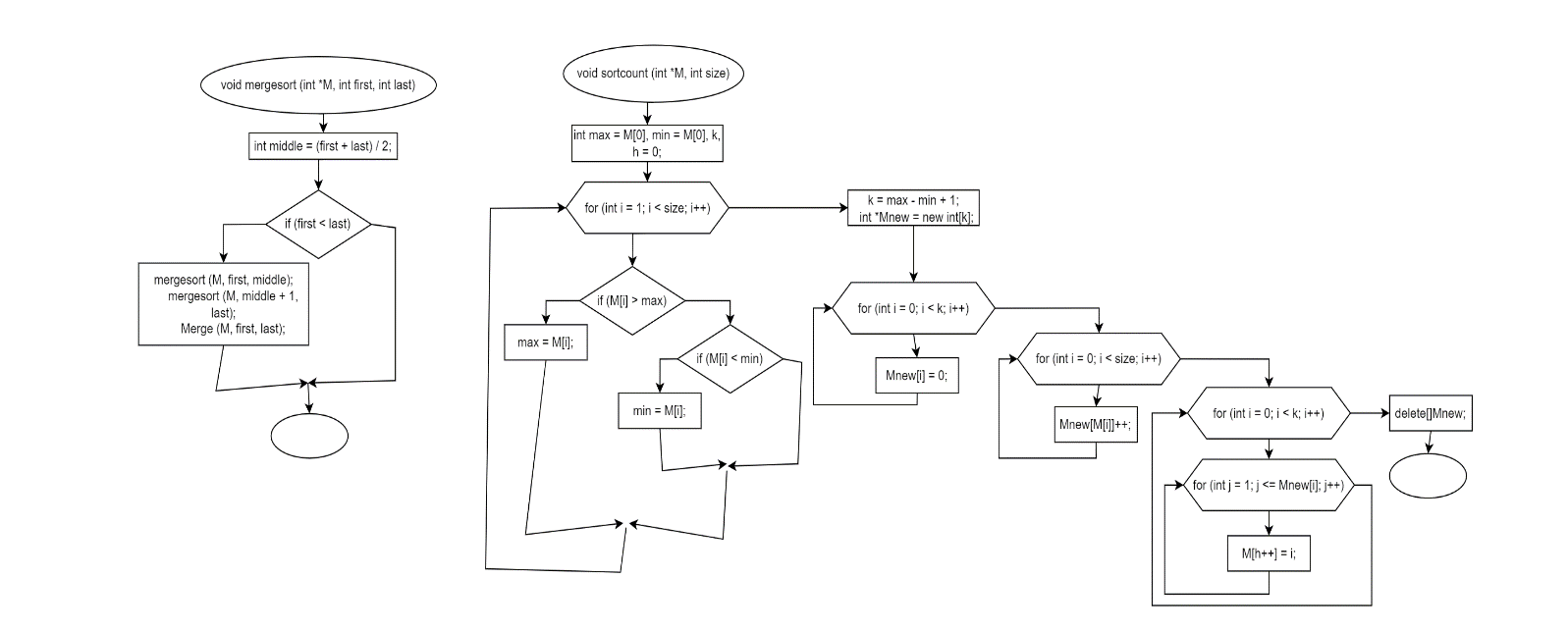
**Анализ решения**

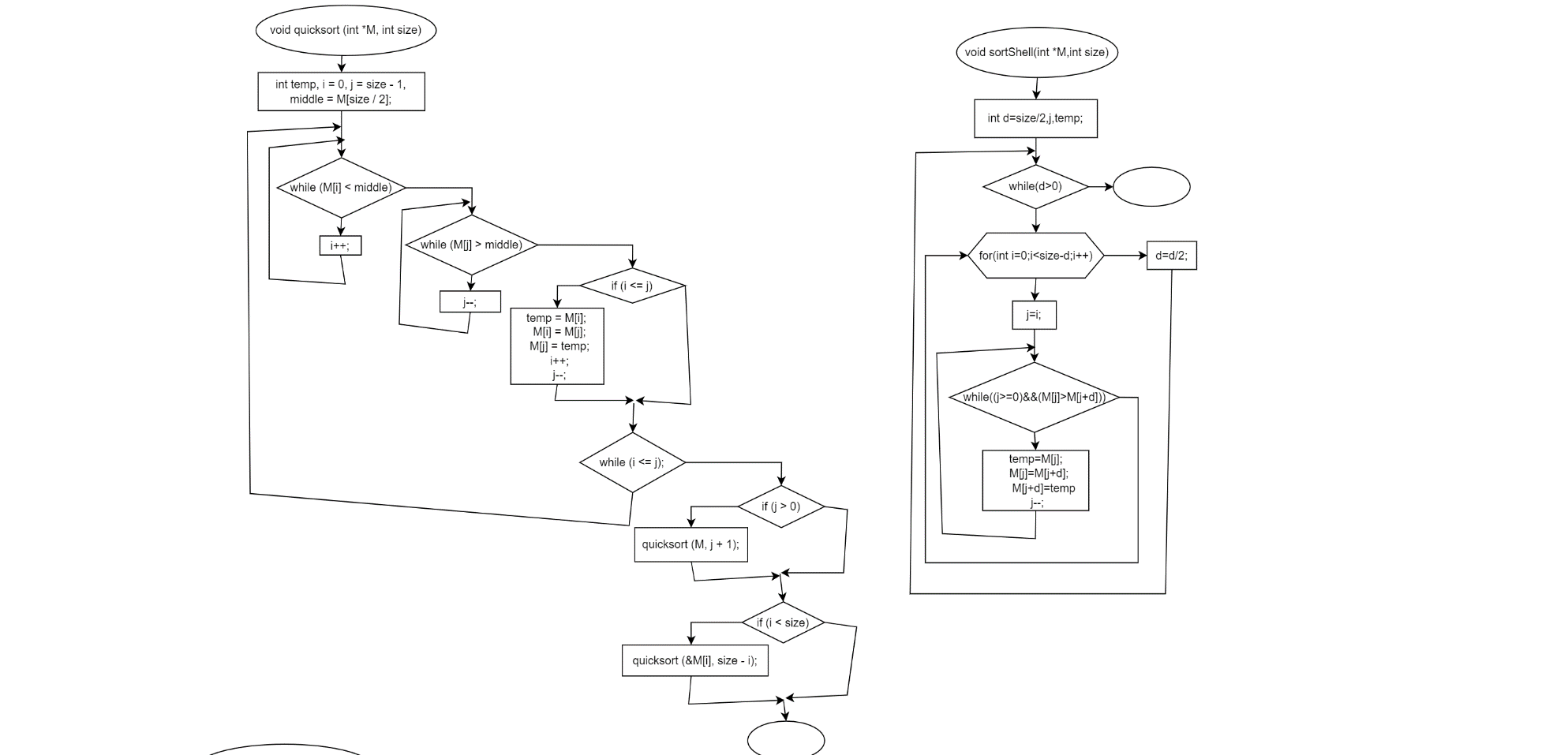
Каждую сортировку засунуть в отдельную функцию. А выбирать сортировки через switch.

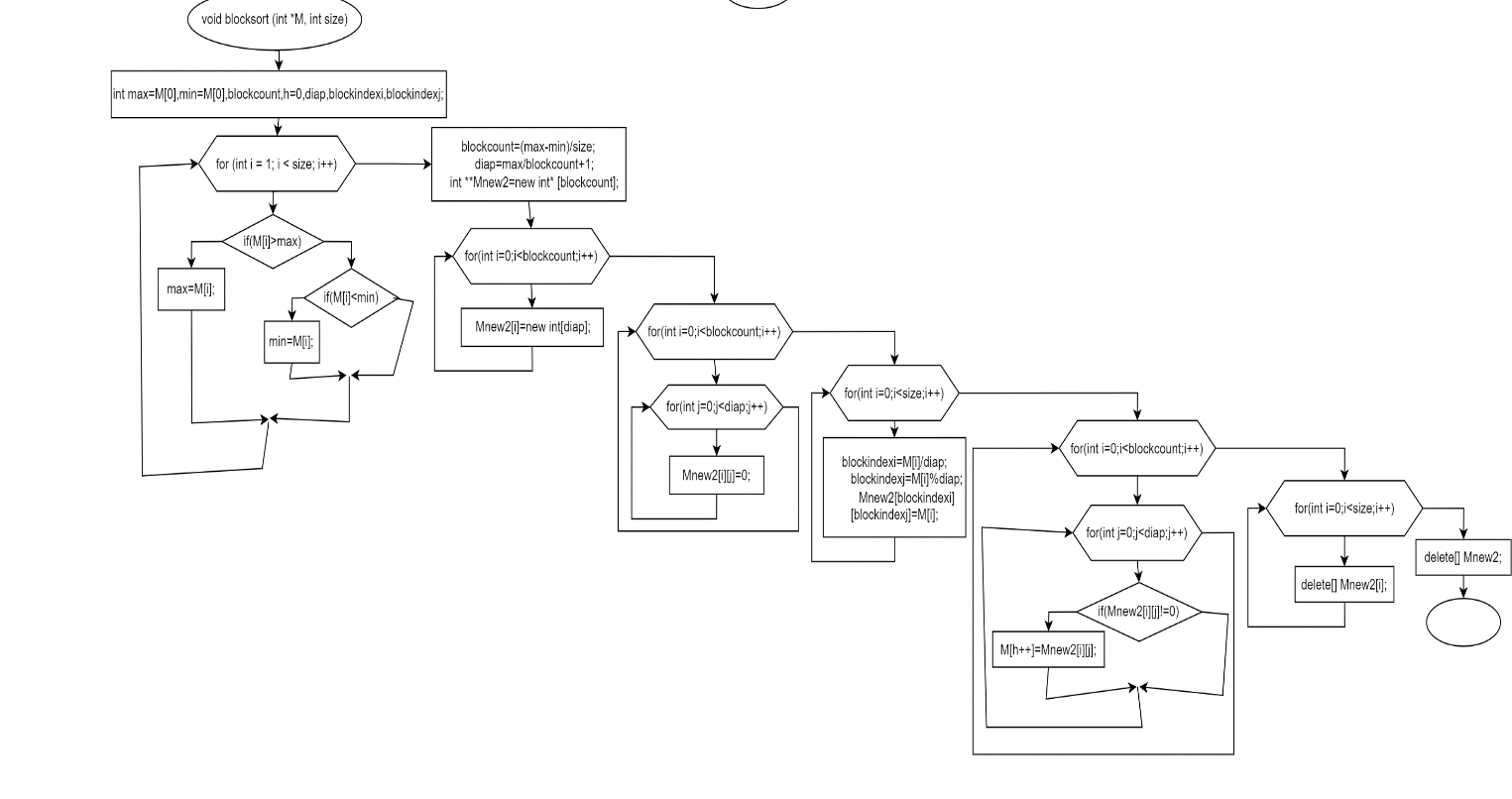
**Алгоритм программы**

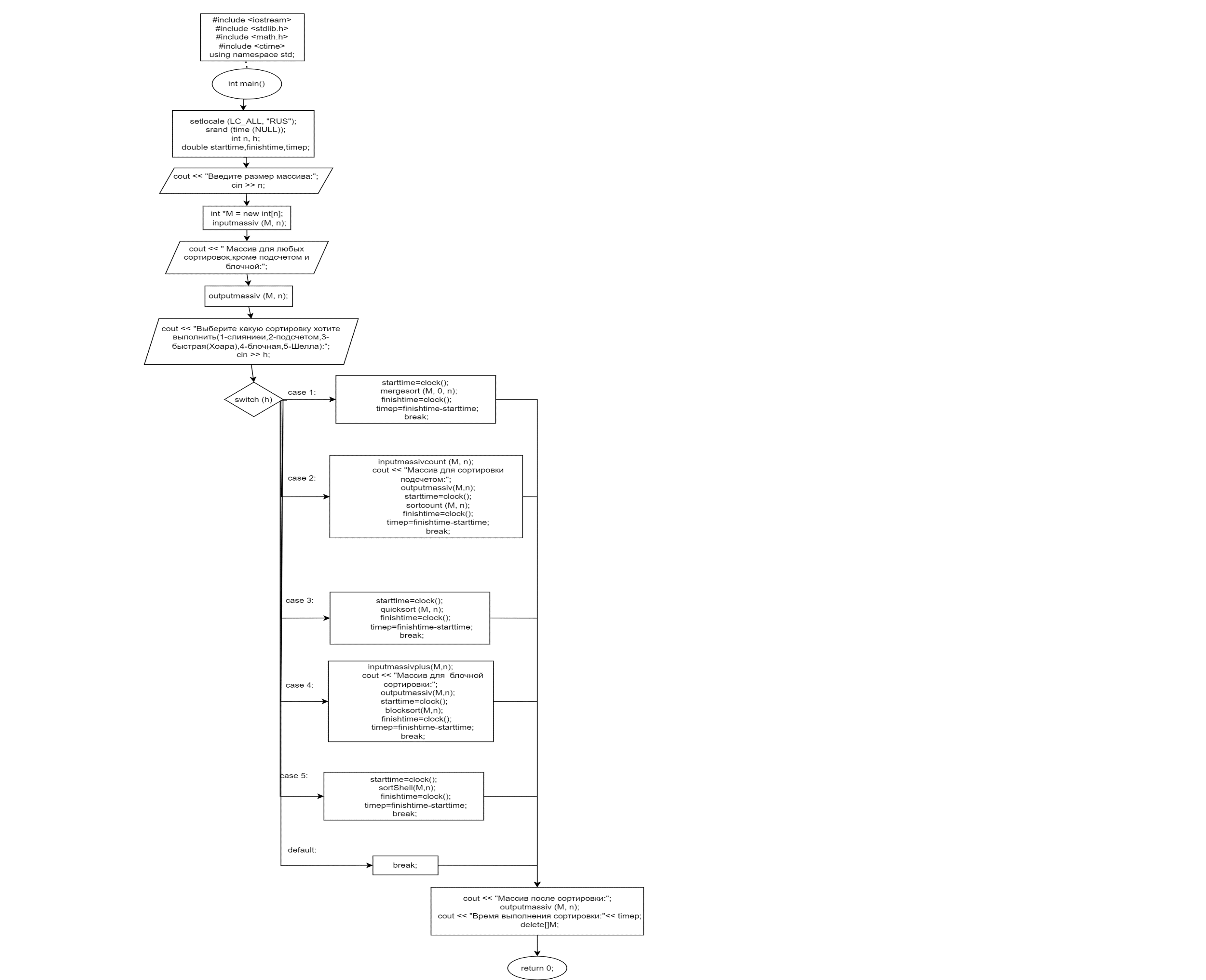


****

****

****

****

****

Код программы:

#include <iostream>

#include <stdlib.h>

#include <math.h>

#include <ctime>

using namespace std;

int\* inputmassiv (int \*M, int size)

{

for (int i = 0; i < size; i++)

{

M[i] = rand () % 101 - 50;

}

return M;

}

int\* inputmassivcount (int \*M, int size)

{

for (int i = 0; i < size; i++)

{

M[i]=rand()%5+0;

}

return M;

}

int\* inputmassivplus (int \*M, int size)

{

for (int i = 0; i < size; i++)

{

M[i]=rand()%101+0;

}

return M;

}

void outputmassiv (int \*M, int size)

{

for (int i = 0; i < size; i++)

{

cout << M[i] << " ";

}

cout << endl;

}

void Merge (int \*M, int first, int last)

{

int middle = (first + last) / 2, start = first, finish = middle + 1;

int \*Mnew = new int[last];

for (int i = first; i < last; i++)

{

if ((start <= middle) && ((finish > last) || (M[start] < M[finish])))

{

Mnew[i] = M[start];

start++;

}

else

{

Mnew[i] = M[finish];

finish++;

}

}

for (int i = first; i < last; i++)

{

M[i] = Mnew[i];

}

delete[]Mnew;

}

void mergesort (int \*M, int first, int last)

{

int middle = (first + last) / 2;

if (first < last)

{

mergesort (M, first, middle);

mergesort (M, middle + 1, last);

Merge (M, first, last);

}

}

void sortcount (int \*M, int size)

{

int max = M[0], min = M[0], k, h = 0;

for (int i = 1; i < size; i++)

{

if (M[i] > max)

{

max = M[i];

}

else

{

if (M[i] < min)

{

min = M[i];

}

}

}

k = max - min + 1;

int \*Mnew = new int[k];

for (int i = 0; i < k; i++)

{

Mnew[i] = 0;

}

for (int i = 0; i < size; i++)

{

Mnew[M[i]]++;

}

for (int i = 0; i < k; i++)

{

for (int j = 1; j <= Mnew[i]; j++)

{

M[h++] = i;

}

}

delete[]Mnew;

}

void quicksort (int \*M, int size)

{

int temp, i = 0, j = size - 1, middle = M[size / 2];

do

{

while (M[i] < middle)

{

i++;

}

while (M[j] > middle)

{

j--;

}

if (i <= j)

{

temp = M[i];

M[i] = M[j];

M[j] = temp;

i++;

j--;

}

}

while (i <= j);

if (j > 0)

{

quicksort (M, j + 1);

}

if (i < size)

{

quicksort (&M[i], size - i);

}

}

void blocksort (int \*M, int size)

{

int max=M[0],min=M[0],blockcount,h=0,diap,blockindexi,blockindexj;

for (int i = 1; i < size; i++)

{

if(M[i]>max)

{

max=M[i];

}

else

{

if(M[i]<min)

{

min=M[i];

}

}

}

blockcount=(max-min)/size;

diap=max/blockcount+1;

int \*\*Mnew2=new int\* [blockcount];

for(int i=0;i<blockcount;i++)

{

Mnew2[i]=new int[diap];

}

for(int i=0;i<blockcount;i++)

{

for(int j=0;j<diap;j++)

{

Mnew2[i][j]=0;

}

}

for(int i=0;i<size;i++)

{

blockindexi=M[i]/diap;

blockindexj=M[i]%diap;

Mnew2[blockindexi][blockindexj]=M[i];

}

for(int i=0;i<blockcount;i++)

{

for(int j=0;j<diap;j++)

{

if(Mnew2[i][j]!=0)

{

M[h++]=Mnew2[i][j];

}

}

}

for(int i=0;i<size;i++)

{

delete[] Mnew2[i];

}

delete[] Mnew2;

}

void sortShell(int \*M,int size)

{

int d=size/2,j,temp;

while(d>0)

{

for(int i=0;i<size-d;i++)

{

j=i;

while((j>=0)&&(M[j]>M[j+d]))

{

temp=M[j];

M[j]=M[j+d];

M[j+d]=temp;

j--;

}

}

d=d/2;

}

}

int main()

{

setlocale (LC\_ALL, "RUS");

srand (time (NULL));

int n, h;

double starttime,finishtime,timep;

cout << "Введите размер массива:";

cin >> n;

int \*M = new int[n];

inputmassiv (M, n);

cout << " Массив для любых сортировок,кроме подсчетом и блочной:";

outputmassiv (M, n);

cout << "Выберите какую сортировку хотите выполнить(1-слияниеи,2-подсчетом,3-быстрая(Хоара),4-блочная,5-Шелла):";

cin >> h;

switch (h)

{

case 1:

{

starttime=clock();

mergesort (M, 0, n);

finishtime=clock();

timep=finishtime-starttime;

break;

}

case 2:

{

inputmassivcount (M, n);

cout << "Массив для сортировки подсчетом:";

outputmassiv(M,n);

starttime=clock();

sortcount (M, n);

finishtime=clock();

timep=finishtime-starttime;

break;

}

case 3:

{

starttime=clock();

quicksort (M, n);

finishtime=clock();

timep=finishtime-starttime;

break;

}

case 4:

{

inputmassivplus(M,n);

cout << "Массив для блочной сортировки:";

outputmassiv(M,n);

starttime=clock();

blocksort(M,n);

finishtime=clock();

timep=finishtime-starttime;

break;

}

case 5:

{

starttime=clock();

sortShell(M,n);

finishtime=clock();

timep=finishtime-starttime;

break;

}

default:

{

break;

}

}

cout << "Массив после сортировки:";

outputmassiv (M, n);

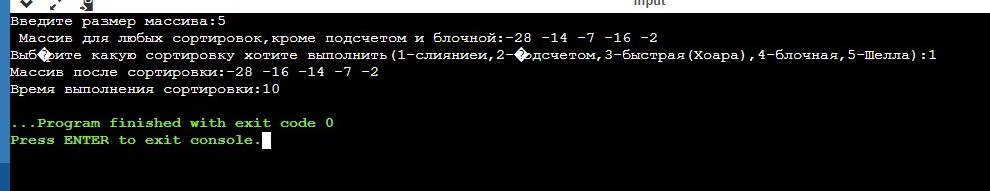
cout << "Время выполнения сортировки:"<< timep;

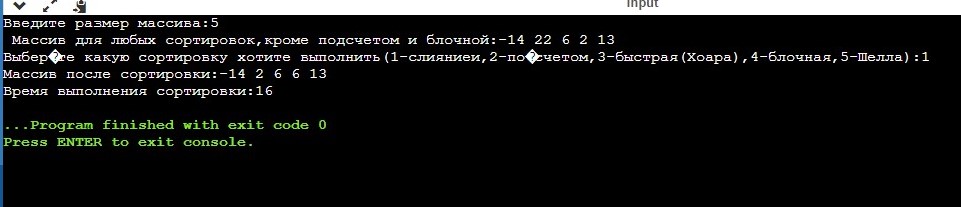
delete[]M;

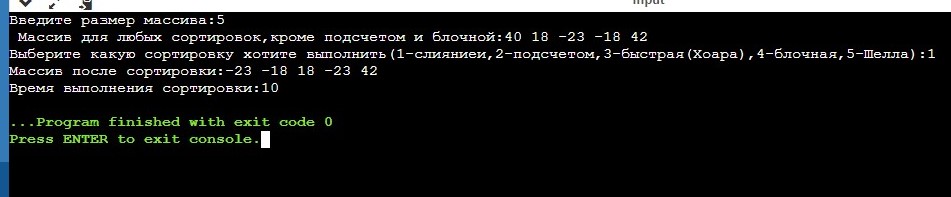
return 0;

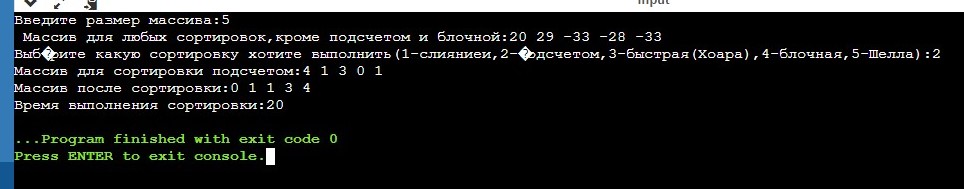
}

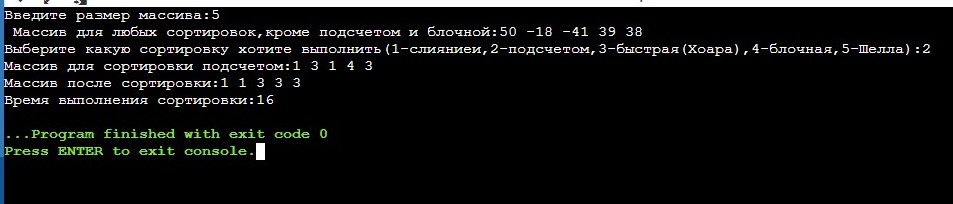
**Результаты работы программы**

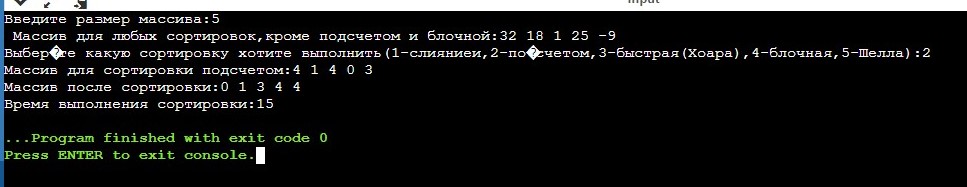


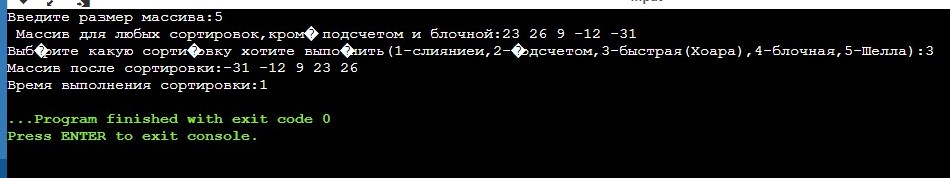


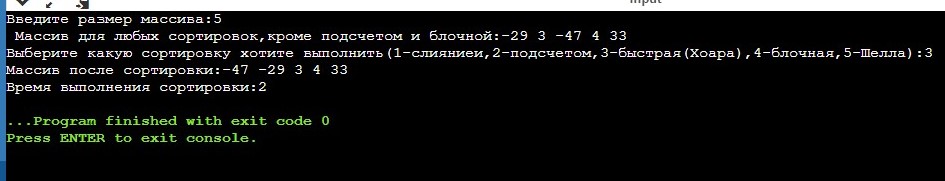


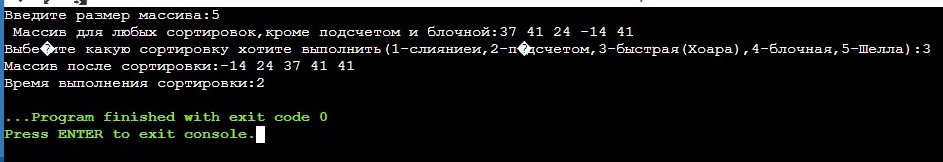


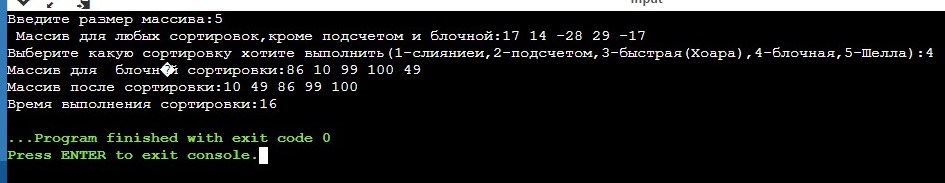


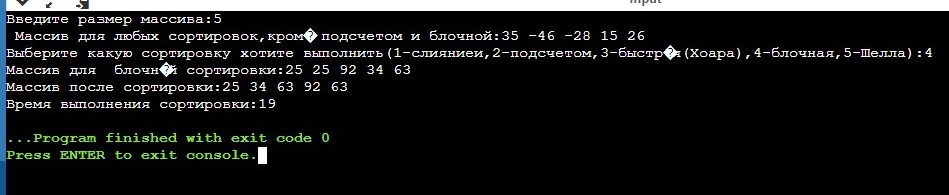


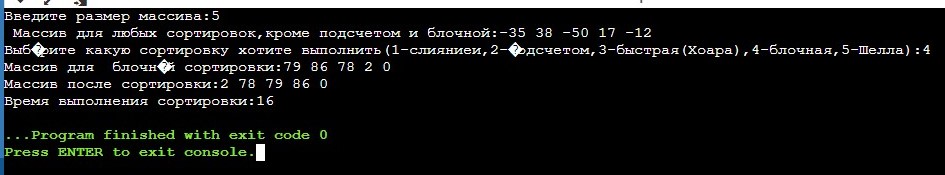


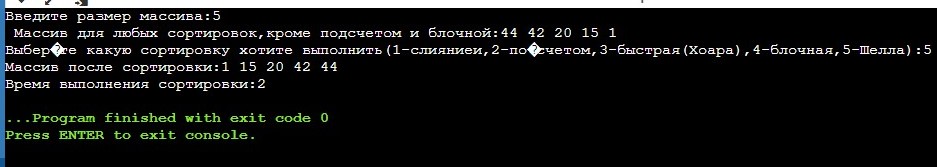


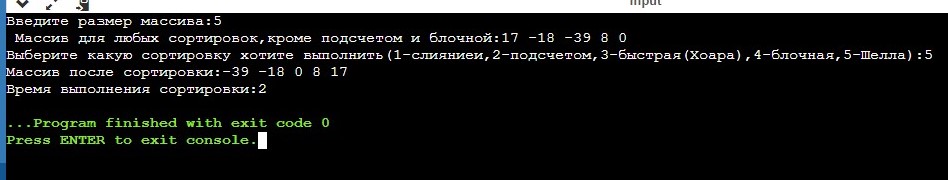


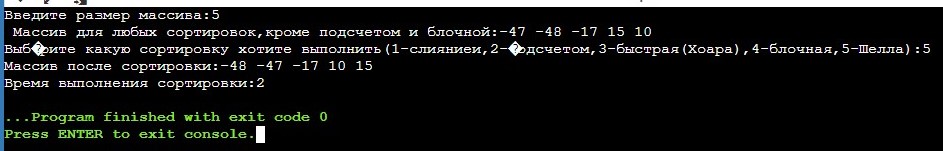












**Анализ результатов**

Как видно из результатов работы программы, среднее время выполнения сортировки слиянием-12 с, среднее время выполнения сортировки подсчетом-17 с, среднее время выполнения быстрой сортировки(Хоара)-1,7 или 2 с, среднее время выполнения блочной сортировки-17 с, среднее время выполнения сортировки Шелла-2 с. Самая быстрые сортировки Шелла и Хоара.